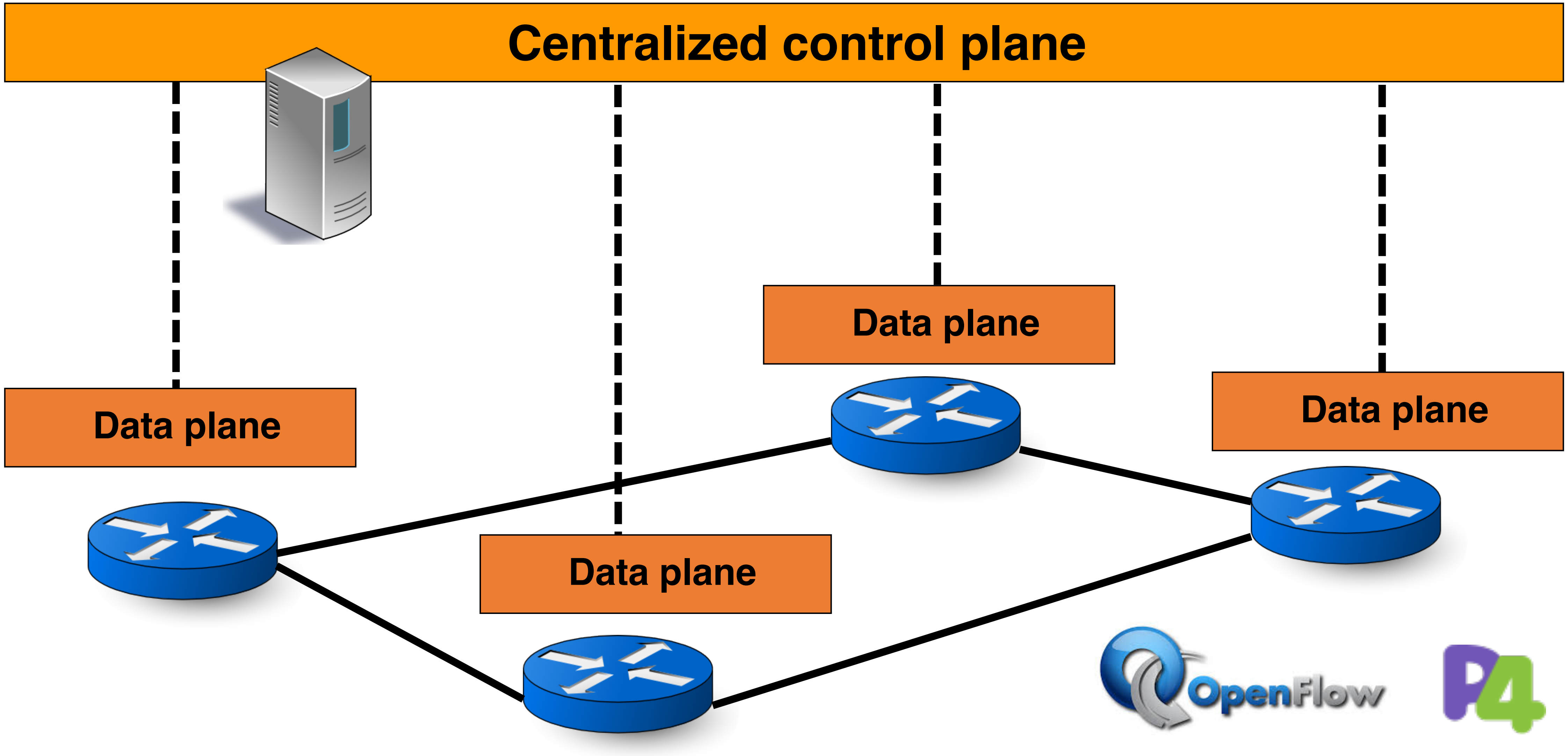
Software Switches

Lecture 10, Computer Networks (198:552)

Fall 2019

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Software-Defined Network (SDN)



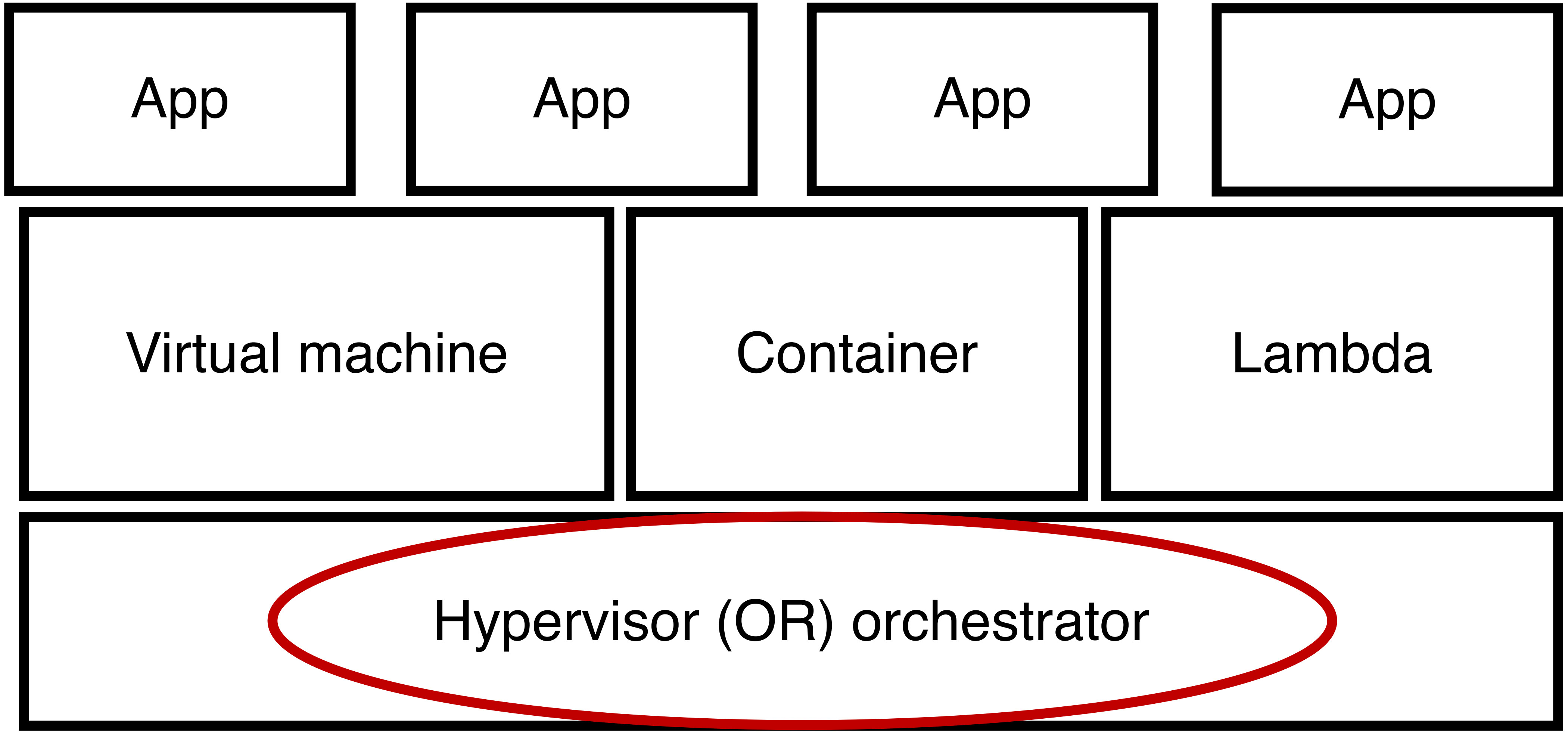
Why software switches?

- Early roots in networking: first switches were fully in software
  - Until high link speeds forced everyone to make ASICs
- As a tool for experimentation with new protocols (eg: Openflow)
- Advent of virtualization
  - Need flexible policies (ie: flow rules) inside endpoints!
  - What policies?

Policies in virtualized switches

- Tenant policies
  - Network virtualization: I want the physical network to look like my own, and nobody else is on it
- Provider policies
  - Traffic must follow the ACLs and paths set by the provider
- Topology “traversal”
  - Use the core of the DCN as a mesh of point to point tunnels

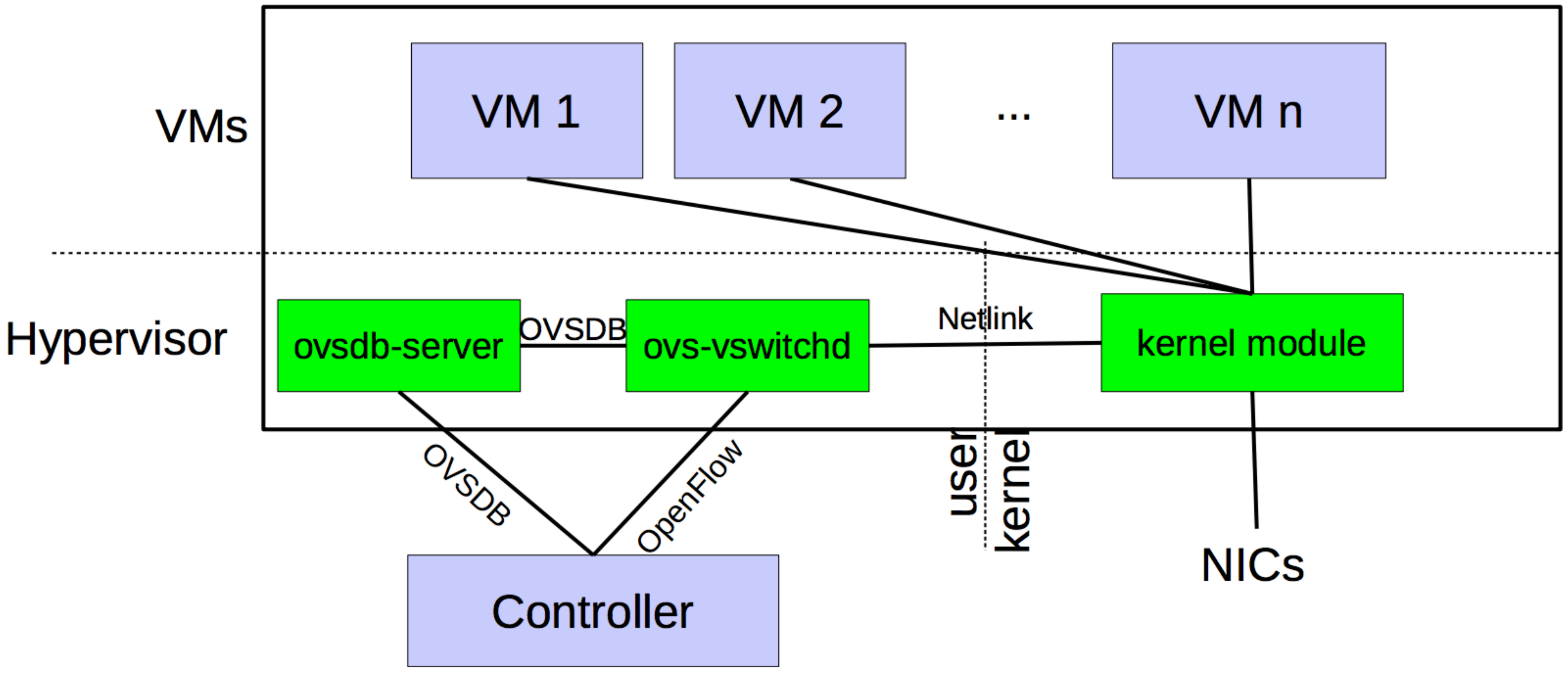
Where should policies be implemented?



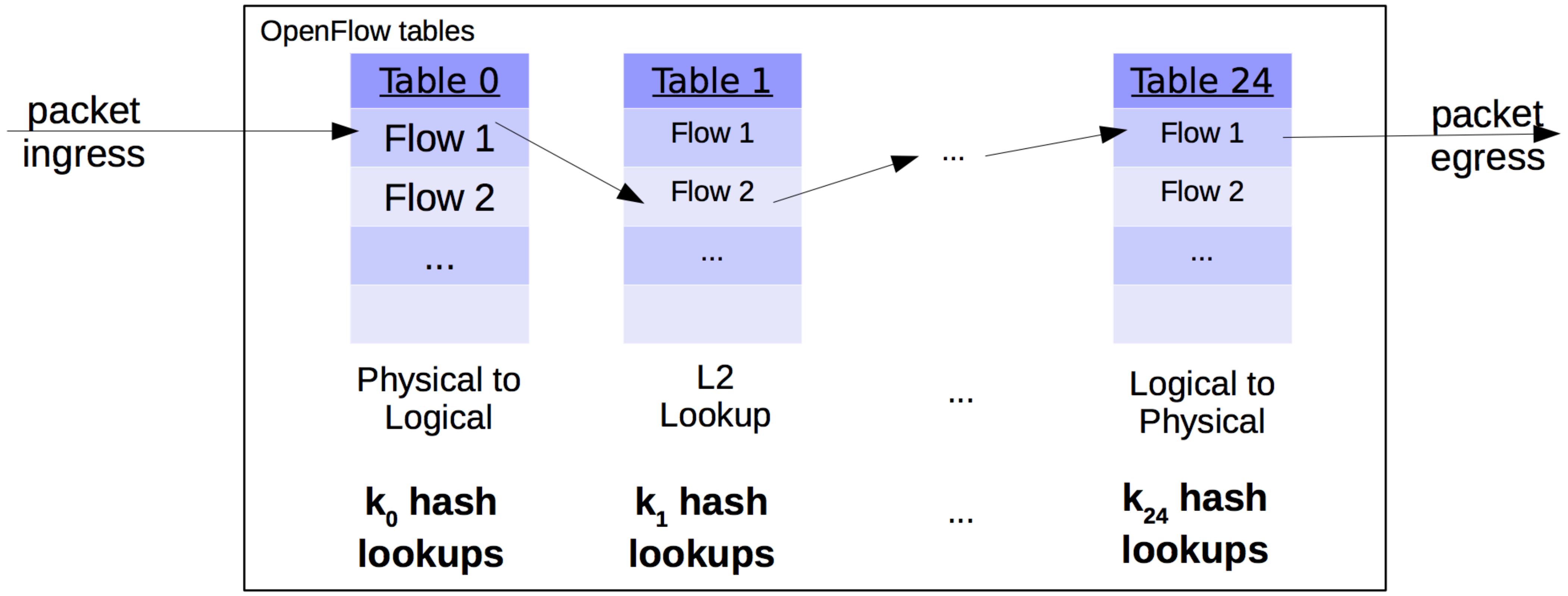
OpenVSwitch: Requirements

- Support large and complex policies
- Support updates in such policies
    - Q: why?
- Allow the hypervisor/orchestrator to support user workloads
    - Don’t take up too much resources
- Process packets with high performance
    - Provide high throughput and low delay

OVS design


VMs
VM 1
VM 2
...
VM n
Hypervisor
ovsdb-server
OVSDB
ovs-vswitchd
Netlink
kernel module
OVSDB
OpenFlow
user
kernel
NICs
Controller

First design: put OF tables in the kernel



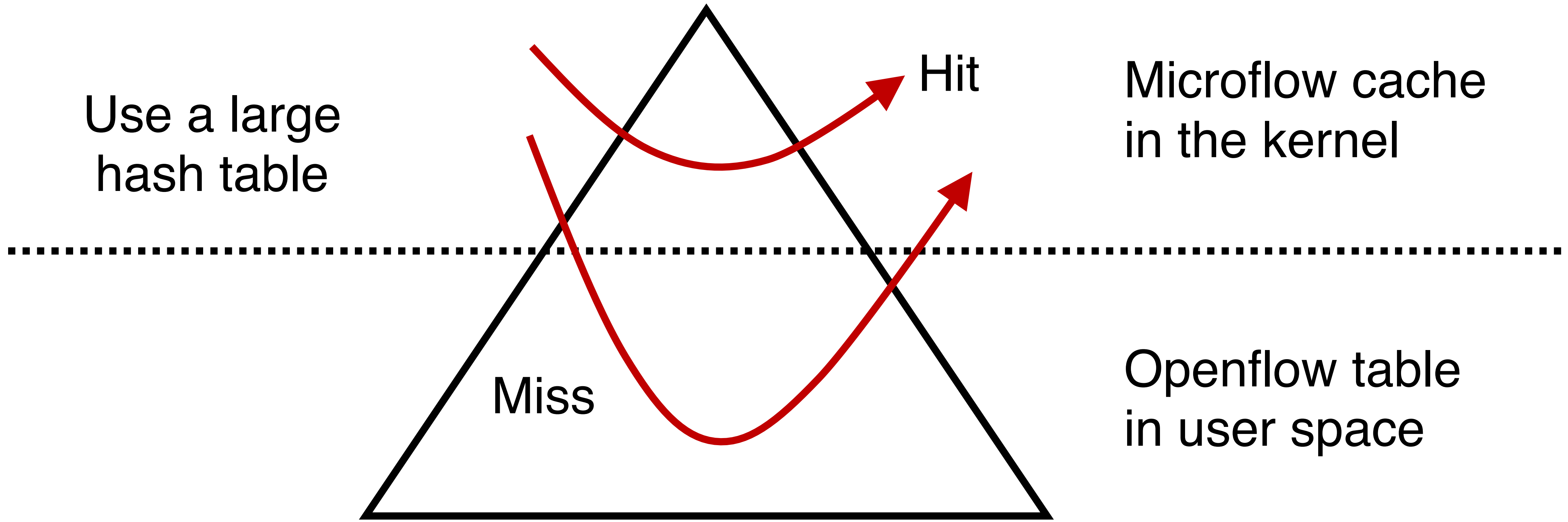
Large policies: Low performance with 100+ lookups per packet

Merging policies is problematic: cross-product explosion

Complex logic in kernel: rules with wildcards require complex algos

Idea 1: Microflow cache

- Microflow: complete set of OF packet headers and metadata
  - Example: srcIP, dstIP, IP TTL, srcMAC, dstMAC
- Use tuple space search to do one lookup per packet



Problems with micro-flows

- Too many micro-flows: e.g., each TCP port
- Many micro-flows may be short lived!
    - Poor cache-hit rate

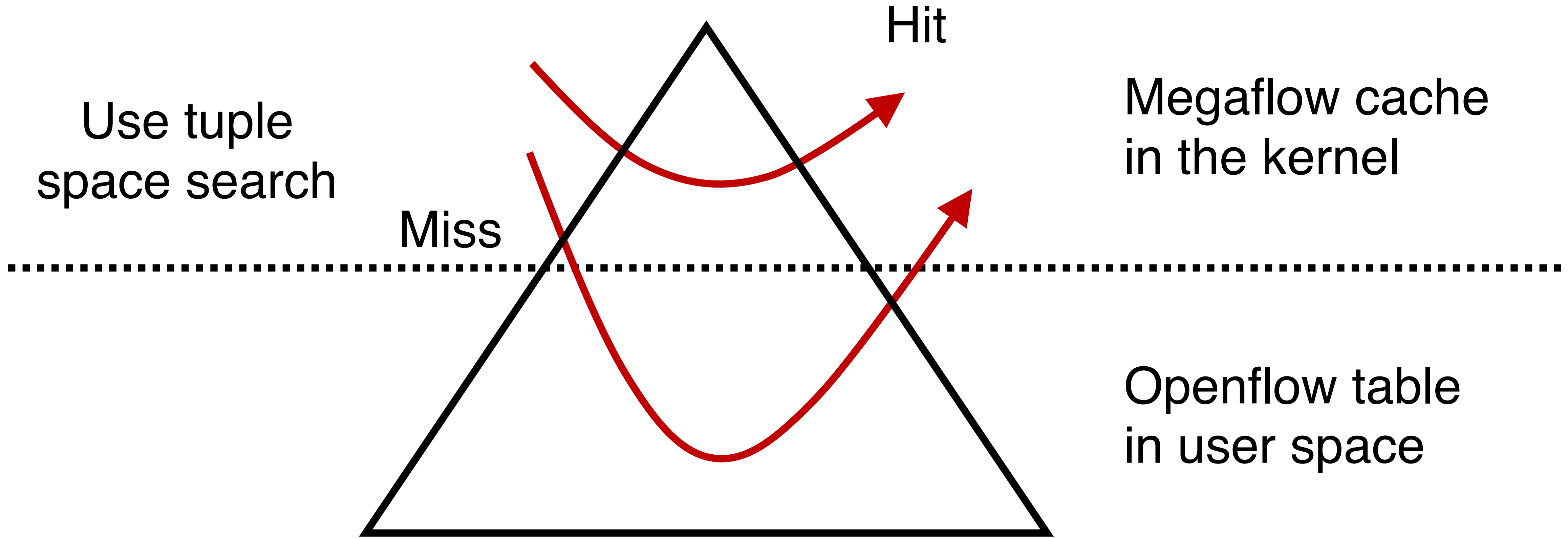
- Can we cache the outcome of rule lookup directly?

- Naive approach: Cross-product explosion!
    - Example: Table 1 on source IP, table 2 on destination IP

- Recurring theme in this paper: avoid up-front (proactive) costs

Idea 2: Mega-flow cache

- Build the cache of rules lazily using just the fields accessed
    - Ex: contain just src/dst IP combinations that appeared in packets



Improvements to mega-flow caching

- You have an OF table. What happens if you populate the mega-flow blindly by concatenating the fields accessed on lookup?
    - Hint 1: consider flow tables that match on highly variable fields
    - Hint 2: consider priorities of rules with overlapping matches
    - Hint 3: consider the number of lookups vs. microflow cache

- OVS introduced a series of *algorithmic* improvements
    - Priority sorting of mega-flow tables
    - Staged lookups starting with more static sets of fields
    - Prefix trie to detect non-overlapping longest-prefix matches
    - Combine with micro-flow cache!

Discussion of OVS

What you said

FAQs

#1 How do megaflows handle wildcards in OpenFlow rules?

FAQs

#2.1 What are the actual sizes of the microflow and megaflow caches?

#2.2 Do the caches overlap with the L1, L2, … caches of the machine?

FAQs

#3.1 What exactly is the overhead of passing a packet between the kernel and the user space process?

#3.2 (How) Do the benefits of OVS caching carry over to user space networking?

FAQs

#4.1 With OVS, are we limited to Openflow? Can we use more flexible protocols?

#4.2 How flexible and perf-critical is OVSDB?

Missing details in perf evaluation

if the paper gave more information on how often in their benchmark system the real locations of virtual switches or their hypervisors were moved. This is because moving very often would inevitably incur overheads due to memory copying and cache misses.

Michael Wu

Missing details in perf evaluation

The author just experimented the effect for each optimization only and the whole combination. They did not take the combination effect of two optimization or three optimization into consideration. I think some combination of two optimizations may achieve better performance and throughput.

Weizhi Wang

Missing details in perf evaluation

Is there any downside on moving the IP tunneled packets transportation virtual? Namely, will the possibility of losing packets increase? [...] Wouldn't connecting to virtual datacenters cost more time, or can it be overlooked?

Xiaochang Chen

Deeper design issues

I feel it possible for one VM to use some techniques of reverse engineering like measuring the time of connection setup to sniff the connections of other vms, similar to Phantom and Sphere's hack to CPU caching. Though I am not quite sure, since Open Vswitch is fully open-sourced, its hash function, seeds are open to all people, it's totally possible to monitor the activities of other VMs in one VM.

Siwei Feng

Improving performance

Another improvement that could be made would be specifying this idea for specific data centers and hardware. While OVS is made to serve data centers in general, research should be done to see how caches should be implemented on a hardware level to best serve the processes outlined in this paper.

Richard Fernandes

Is it possible to offload the flow table search and flow computation to some special hardware, e.g., FPGA?

Qiongwen Xu

Improving performance

staged lookup [...] statically divides field into four groups. A
better choice is to refer to importance of different headers in different
scenes adaptively.

Huixiong Qin